MON-69

SMOKE SIGNAL
Chieftain Computers

31336 Via Colinas
Westlake Village, California 91362

(213) 889-9340

MON-69

SMOKE SIGNAL BROADCASTING

6809 RESIDENT MONITOR

TABLE OF CONTENTS

APPENDICES

I. GETTING STARTED

This manual is intended to cover all the 6809 monitors available from SSB for the various disk controllers and cpu boards. The supported monitors are:

MON69 for the SCB-69 cpu and BFD-68 single density controller
MON09 for the SWT 6809 cpu and BFD-68 single density controller
MON69D for the SCB-69 cpu and DCB-4 double density controller
MON09D for the SWT 6809 cpu and DCB-4 double density controller

This manual is intended to cover the essential items of the 6809 monitors and disk drivers used by SSB. Source listings and source-on-disk are available from SSB separately.

For those of you who are anxious to get started, brief directions are given here. For the rest of you, please feel free to read the manual, and return here for startup instructions.

If the monitor ROM is not already installed, follow normal static prevention procedures and install it in the F800-FFFF socket on the CPU board. Make sure that a serial interface card for the terminal is installed in I/O slot 2 (address F7E8 for Chieftain systems, E008 for SWT systems). Also be sure that the baud clock generator is set for a times 64 clock for Chieftain users (times 16 for others). Also be sure that the system contains at least 4K of RAM physically addressed to include the D000-DFFF address space. Finally, check to see that the I/O space is located at F780-F7FF for Chieftain systems, or E000-E01F for SWT systems. Apply power, and the system will sign on with a <CR><LF>* sequence. Refer to Appendix A for a summary of the commands available, or see section III for a detailed description of the commands.

NOTE: For SSB disk system users, refer to Appendix D for appropriate instructions on installing disk controllers.

Now that you have your system running, please read the remainder of this manual to discover all the things that the monitor will do for you.

II. PHILOSOPHY

Underlying the development of this monitor program is the philosophy that the ROM monitor should include the functions necessary to get the system cold started, provide for lowest level debugging, and contain the fixed routines required by the processor; all other functions should be handled by the operating system software, which is typically disk based. This philosophy permits the monitor ROM to remain essentially unchanged as the system software becomes ever more powerful through evolution.

The monitor ROM includes within itself the firmware for disk drivers and a cold start boot routine for the SSB disk system. These must be in ROM, or else would have to be keyed in by hand after each power interruption! Placing these routines in ROM essentially freezes the characteristics of the disk drivers, and provides a common place where the disk file manager and other system software can access the low level drivers, in addition to providing a virtually automatic cold start boot capability.

The monitor is supplied in EPROM format to provide for easy customization. Source listings of the appropriate monitor can be obtained from SSB. Contact the sales department for current price information.

III. COMMANDS

All 6809 monitors from SSB contain a basic set of commands. Where differences occur, a detailed description of the differences will appear.

All keyboard commands consist of a single alphabetic character, possibly followed by one or more arguments. A valid command will be acknowledged by the system outputting a space after the command letter; in some cases, such as "clear breakpoints", the system will supply <CR><LF>* so soon after the space that you will not be able to tell that anything happened.

Any of the commands may be aborted by typing a non-hex character (such as <CR>) where a hex number is expected. Additionally, the commands that are capable of producing lengthy output may be aborted by typing a <CR> on the keyboard; this will be recognized at the end of the line of output, and the system will return to command level.

COMMAND DESCRIPTIONS:

A - Examine and perhaps change the A accumulator value on the stack. This command will not function if the E bit on the stack is cleared (see the 6809 programming manual for details of the functions of the bits in the CC register on the stack as a result of an interrupt). Typing the letter A will cause the contents of the stacked A register to be displayed. If you wish to change this value, type a new 2 character hex value after the old value is displayed. Otherwise, type a carriage return to return to command level.

B - Examine and perhaps change the B accumulator value on the stack. This command will not work if the E bit on the stack is cleared.

C - Examine and perhaps change the CC register on the stack. The MSB of this register is the E bit; this bit controls the accessibility of all other registers (except PC) on the stack.

D - Examine and perhaps change the DP register on the stack. This command will not function if the E bit on the stack is cleared.

E - (not used)

F - Find a byte string in the specified portion of memory. The syntax is F <starting addr> <ending addr> <first byte value> <second byte value> <third byte value> <fourth byte value>. The second, third, and fourth byte values are optional; use <CR> instead of the byte value if you wish to search for a shorter string. The monitor will display each address in the specified range where the byte string exists. (Available only with MON69/MON09 [single density monitors])

G - Go to user's job on stack. This causes the execution of an RTI instruction, thereby resuming execution of the last code interrupted. The state of the E bit on the stack will determine which registers are pulled from the stack. Note: if the CC bits for either I or F are clear, the interrupt(s) become(s) enabled when this command is executed.

H - (not used)

I - Initialize memory. The syntax is I <starting addr> <ending addr> <byte value>. Execution of this command causes memory in the specified range to be set to the specified byte value. This is especially handy for initializing RAM to 3F (SWI) before loading the program to be debugged.

J - Jump to specified address. The syntax is J <address>. Note: this command executes a JSR to the specified address. If the code being executed ends with an RTS and did not change the stack pointer, control will return to the monitor. This characteristic can be used to test subroutines during debug.

K - Set a breakpoint. The syntax is K <address>. As many as six breakpoints may be set at once. If more than six are attempted, the monitor responds "FULL" and ignores the request. Breakpoints are automatically cleared when encountered. Each time a breakpoint is cleared, its space becomes available to the K command.

Breakpoints should only be set in RAM, since the K command operates by saving the byte at the specified address, and replacing it with a SWI instruction.

When execution resumes (usually due to the G command) and a SWI is encountered, it is checked to see if it is a breakpoint. If so, a "#" is displayed to acknowledge that the SWI is from a breakpoint. If not, no "#" is printed, and no breakpoint clearing action occurs. In either case, the program counter on the stack is adjusted to point to the location where the SWI was found. This allows the G command to continue execution from this point if the SWI were due to a breakpoint. If the SWI was not set by the K command, then the PC will still be at the SWI, and typing G merely repeats the execution of the SWI.

Note: the monitor initializes the SWI secondary vector to point to the breakpoint routine whenever a hardware reset occurs; however, this vector is available to the user and can be directed away from the breakpoint handler. Therefore, it is possible (though unlikely) that a wayward program could alter this vector, causing unpredictable results.

[CTL K] - Kill the currently active breakpoints. No argument is required, and no acknowledgement is provided. This routine is automatically invoked by a reset, and by numerous returns to the command level.

L - Load tape. This command takes as an argument a checksummed nibble organized character string in the same format as used by the MIKBUG (Motorola TM) L command. (This format is produced by the P command in this monitor.) The characters S9 at the beginning of a block will terminate the loading process, as will a checksum error.

(Available only with MON69/MON09 [single density monitors])

M - Memory examine and perhaps change. The syntax is M <address>. Using this command will open the specified location and display its contents. To change the contents,

type in valid hex characters. To open the preceeding location, type '^' (circumflex). To open the indirect location <address> and <address+1> type '.'. To open the indirect location <address-1> and <address> type '@'. These last two forms are useful in tracing indirect addressing. To open the next location, type any character that is not <CR>, a hex digit, or any of the above listed opening characters (space is a convenient character for this purpose). To close the location without change, type <CR>. Whenever a location is changed, the new value is read back to see if it stored OK. If not, a question mark is printed after the new value, and the next location is opened.

N - (not used)

O - (not used)

P - Punch tape command. The syntax is P <start address> and <end address>. The monitor outputs a MIKBUG (Motorola TM) compatible character string to the terminal. Device control codes are supplied, facilitating automatic tape punching when used with certain terminals. The format used by this command is compatible with that expected by the L command. This command, as well as others that are capable of producing lenghty of producing lengthy output, can be aborted by output, can be aborted by typing <CR> on the terminal; at the end of each line of output, the keyboard is checked and, if a <CR> has been typed, the command is aborted.

(Available only with MON69/MON09 [single density monitors])

Q - Quick start. No argument is required. This command invokes the cold start bootstrap loader routine contained within the monitor (see Appendix E).

R - Register display, No argument is required. This command displays a title line, then displays the contents of the registers on the stack. The condition code register is displayed twice, once in hex format and once at the end of the line as the bits that are set within it. If the E bit is clear, then only the CC and PC registers will be displayed, although all the registers are titled. The value displayed for the SP register is the address of the location of the CC register.

S - Speedy register display. This command is identical to the R command, except that no titles are displayed. This is convenient when working with a slow terminal.

T - Text input to memory. The syntax is T <start address> <text string> <control D>. Any character (including <null> and <CR>) except control D can be input to memory. This command is very useful for altering text strings in programs and in the file name portion of file control blocks.

(Available only with MON69/MON09 [single density monitors])

U - Examine and perhaps change the value in the user stack register on the system stack. To change the value, type a new 4 hex digit number. To leave the old value unchanged, type <CR>. This command will not function if the E bit on the stack is cleared.

V - Examine and perhaps change the value in the program counter register on the stack. Operation is the same as with the U command, except that this command always functions, regardless of the state of the E bit.

W - Warm start into DOS. No argument is required. This command transfers control to the warm start address of DOS.

NOTE: If DOS has been over written or has not been loaded into memory already, this command should not be used.

X - Examine and perhaps change the value in the X register on the stack. Operation is the same as with the U command.

Y - Examine and perhaps change the value in the Y register on the stack. Operation is the same as with the U command.

Z - Formatted memory dump to the terminal. The syntax is Z <start address> <end address>. As with other commands capable of producing lengthy ouput, the command may be aborted by typing <CR>. This command forces the supplied start and end addresses to mod 16 boundaries such that each line displayed will start at address XXX0 and end at address YYYF. The data in these addresses is displayed first in hexadecimal format, and then interpreted as ASCII. Non-printing ASCII characters are replaced with a dot.

. - Reopen last location examined by the M command. This command is identical to typing M <last address opened>. It is especially useful during debug sessions where the contents of a memory location are inspected, the next section of program code is executed, and then the same memory location needs to be inspected again. Typing dot will reopen the location.

8 - Transfer program control to $E800. This command is used to jump to a program whose starting address is $E800.

(Available only with MON69D-2/MON09D-2 [double density monitors])

IV. INTERRUPT HANDLING

RESET

Although reset is not strictly an interrupt, it is handled by the processor in much the same way as the interrupts, and therefore is described in this section.

When the processor executes a reset, control is given to the monitor's reset routine which establishes a memory map, initializes the control port, initializes the interrupt vectors to their default addresses, clears all breakpoints, and proceeds to command level.

NMI

The NMI primary vector points to a jump indirect that transfers control to the address contained in the secondary NMI vector. This address is set to restart when a reset is executed, thus NMI can be used instead of reset with the added advantage that it does not alter any of the vectors. This is the recommended mode of operation. The only time that NMI will not work as a reset is when the secondary NMI vector has been altered, either intentionally or as a result of a runaway program. User software may wish to change the NMI secondary vector to point into the user program, allowing the NMI function to return control to the user program.

IRQ

The IRQ primary vector points to a jump indirect that transfers control to the address contained in the secondary IRQ vector. This vector is initialized by reset to point to a dummy RTI instruction; thus, IRQs are ignored until the user sets up the secondary IRQ vector.

FIRQ

The FIRQ primary vector points to a jump indirect that transfers control to the address contained in the secondary FIRQ vector. As with IRQ, this vector is initialized to point to a dummy RTI; to make use of FIRQ, user software must set up its secondary vector.

SWI

The SWI primary vector points to a jump indirect that transfers control to the address contained in the secondary SWI vector. After a reset, this vector points to the breakpoint handler (see description of the K command). It is recommended that SWI be used only for breakpoints; however, if desired, the secondary SWI vector is available to user software and may be directed to user code.

SWI2

The SWI2 primary vector points to a jump indirect that transfers control to the address contained in the secondary SWI2 vector. As with IRQ, this vector is initialized to point to a dummy RTI; to make use of SWI2, user software must set up its secondary vector.

SWI3

The SWI3 service routine in the monitor provides a powerful system capability: software interrupt driven system tasks.

After reset, the various registers associated with SWI3 are initialized to cause the SWI3 instruction to transfer control to the address contained in the secondary SWI3 vector.

Whenever a SWI3 is executed, the monitor will increment the program counter value that was saved on the stack. After the saved program counter is incremented, the service origin vector SVCO is checked; if the value here is $FFFF (the default), then all registers except PC are restored, and control is transferred to the address contained in the secondary SWI3 vector. When the user service routine finishes by executing an RTI, control will return to one location past the SWI3 call (the byte following the SWI3 will be skipped).

If the service origin vector does not contain $FFFF, then the value it does contain is used to point to the service table. The byte that follows the SWI3 is multiplied by 2 and used as an offset into the service table; the address found at this location in the service table becomes the destination of the routine. When control is passed to this address, all registers except CC, PC, and U contain the same values that they had at the time of SWI3. (The U register contains the value that should be loaded into the SP before an RTI is finally executed.)

Before passing control to the address, the routine checks to see that the position selected within the table is at or below the service limit vector (SVCL) contents. If so, control is transferred as already described. If not, then processing proceeds to transfer control through the SWI3 secondary vector, just as though the service origin vector had not been set. This characteristic can be used to check for out of bounds calls to the service routine.

The utility of the service table routine is that it allows user software to keep a table of pointers to user routines, and to access any of those routines in a position independent fashion by doing a SWI3 followed by an FCB nn, where nn specifies the nth-1 entry in the service table. See Appendix E for a detailed example.

V. JUMP TABLE (POINTER TABLE)

The beginning of the monitor contains a table of vectors that point to the various available routines in the monitor. A monitor routine should be called with a JMP (or JSR) indirect through the pointer table. The locations of these pointers will remain fixed in future versions of this monitor; the routines themselves may move within the monitor; therefore, they should never be called directly. The function of each of the available routines is explained below. See Appendix B for a list of the registers that are altered by each routine.

F800 MONIT — Perform the same function as a reset: establish a memory map; initialize vectors, stack, control port; clear breakpoints; transfer control to the command processor.

F802 CMD — Prompt for a command letter and process it when it is receivied.

F804 INCH — Get a single character from the terminal without stripping the parity bit or echoing the character; return the character in the A register.

F806 INEEE — Get a single character from the terminal; set the parity bit to 0; echo the character to the terminal; return the character in the A register.

F808 INCHK — Check the keyboard to see if a character is waiting; return Z bit = 1 if so, Z bit = 0 if not.

F80A OUTCH — Output the character in the A register to the terminal.

F80C PDATA — Display the string of characters pointed to by the X register; the string is terminated by a hex 04. On exit, A contains 04, and X points to the location after the terminator.

F80E PCRLF — Output the CRLF string to the terminal. (Extra space is provided in the string for customization, if desired).

F810 PSTR — Display a CRLF, then display the string pointed to by the X register. On exit, A contains 04 and X points to the location after the terminator.

F812 LRA — Load real address. Enter with X containing the logical address to be converted; exits with A containing bits 16-19 of the physical address, and X containing bits 0-15 of the physical address.

F814 RESTRT — Resets the stack pointer; clears all breakpoints, transfer control to command level.

VI. MONITOR RAM USAGE

The monitor uses RAM between approximately F380 and F3FF (DF80 - DFFF). The lower limit is variable because the monitor's stack extends downwards from F3BD (DFBD); if any interrupts, user subroutines, or breakpoints are executed, RAM usage may extend to lower addresses.

Memory from F3BE through F3CF (DFBE - DFCF) is available for change by user software. This region includes all the secondary vectors and the service table vectors. See the INTERRUPT HANDLING section for details on these.

Memory from F3D0 through F3FF (DFD0 - DFFF) is available for inspection by user software, but it must not be changed by user software. Of particular interest is the 16 byte long MAPTBL area. This table stores a copy of the memory map that exists in the mapping RAM on the CPU board, and it is the table that is accessed by LRA when a logical-to-physical address conversion is required. Each byte in the table stores two 4-bit values associated with its logical block address (a block is a 4K region of memory; for example, block 2 includes addresses 2000-2FFF). The least significant 4 bits specify the physical block address associated with the logical block. The most significant 4 bits specify the page on which the physical memory is to be found (a page is one of 16 sets of 64K address spaces). Page F is reserved for the system; this page contains I/O, disk interface, the monitor program, and any other fundamental system devices.

NOTE: The SCB-69 contains an option switch which allows the user to select a dual memory map for the memory area DF80-DFFF. This memory is normally used by DOS69/D and MON69/D for access to the disk driver scratch pad. With the dual memory map option on, all access to the DF80-DFFF area will be translated to the F380-F3FF area. Example: The map table located at DFD0-DFDF in the MON09/D monitor is translated to F3D0-F3DF for the MON69/D monitor. All programs that are to run on both monitors should use the lower memory map as the SWT system can not translate the F3D0-F3DF area to the lower area.

APPENDIX A

COMMAND SET SUMMARY:

```
   A  Examine & change A register
   B  Examine & change B register
   C  Examine & change CC register
   D  Examine & change DP register
   E  (not used)
*  F  Find a byte string
   G  Go to user's program on stack
   H  (not used)
   I  Initialize memory
   J  Jump (subroutine) to addr
   K  Set breakpoint
*  L  Load MIKBUG format tape ***
   M  Memory examine & change
   N  (not used)
   O  (not used)
*  P  Punch MIKBUG format tape ***
   Q  Quick start (disk boot)
   R  Register display
   S  Speedy register display (no titles)
*  T  Text input to memory
   U  Examine & change U register
   V  Examine & change PC register
   W  Warm start into DOS
   X  Examine & change X register
   Y  Examine & change Y register
   Z  Display formatted memory dump
   .  Reopen last address opened in 'M'
   [CTL K] Clear all breakpoints
** 8  JSR to $E800
```

The following commands are usable when a memory location is open in 'M':

```
.  Open the address at M,M+1
@  Open the address at M-1,M
```

These are useful in tracing indirect addressing.

```
  * Available only with MON69/MON09
 ** Available only with MON69D-2/MON09D-2
*** MIKBUG is a registered trademark of Motorola, Inc.
```

APPENDIX B

JUMP TABLE (POINTER TABLE) SUMMARY:

The column titled CABDXYUP indicates whether or not each of the CPU registers was modified by the routine; a dot indicates that the register was modified, while a letter indicates that the register was preserved.

ADDR	CABDXYUP	NAME	FUNCTION
F800	...D....	MONIT	Initialize monitor
F802	...D....	CMD	Command level monitor entry
F804	..BDXYUP	INCH	Get a raw character from terminal
F806	..BDXYUP	INEEE	Get & echo a character from term
F808	.ABDXYUP	INCHK	Check for character waiting
F80A	.ABDXYUP	OUTCH	Output character to terminal
F80C	..BD.YUP	PDATA	Output string to terminal
F80E	..BDXYUP	PCRLF	Output CRLF string to term
F810	..BDXYUP	PSTR	Output CRLF, then string
F812	..BD.YUP	LRA	Convert logical to physical addr
F814	...D....	RESTRT	Monitor restart entry

APPENDIX C

MONITOR RAM USAGE:

MON69	MON09	
???? - F3BD	???? - DFBD	Stack
F3BE - F3C1	DFBE - DFBF	NMI secondary vector
F3C0 - F3C1	DFC0 - DFC1	Reserved
F3C2 - F3C2	DFC2 - DFC3	SWI3 secondary vector
F3C4 - F3C5	DFC4 - DFC5	SWI2 secondary vector
F3C6 - F3C7	DFC6 - DFC7	FIRQ secondary vector
F3C8 - F3C9	DFC8 - DFC9	IRQ secondary vector
F3CA - F3CB	DFCA - DFCB	SWI secondary vector
F3CC - F3CD	DFCC - DFCD	Service table origin
F3CE - F3CF	DFCE - DFCF	Service table top
F3D0 - F3DF	DFD0 - DFDF	Memory mapping table
F3E0 - F3E1	DFE0 - DFE1	Saved stack pointer
F3E2 - F3E3	DFE2 - DFE3	Control port address
F3E4 - F3E5	DFE4 - DFE5	Last addr examined with 'M'
F3E6 - F3ED	DFE6 - DFED	Reserved for disk firmware
F3EE - F3FF	DFEE - DFFF	Breakpoint table

NOTE:With the daul memory map option (switch S1-3) on, the SCB-69 will translate all memory references to the MON09/D RAM area (DF80-DFFF) to the corresponding MON69/D RAM area.

APPENDIX D

MODIFYING SSB DISK CONTROLLER BOARDS:

There are two main disk controller boards available from SSB; the single density BFD-68 controller and the double density DCB-4 controller.

Many single density BFD-68 disk controller boards were supplied for operation at address 9FFC. If you have one of these, it must be modified for operation at F77C. The modification involves rearranging the inputs to the decoders by changing jumpers and/or cutting and jumpering a few traces.

Two types of BFD-68 controller boards have been supplied that are suitable for this conversion; the earlier type had one edge connector at the top of the board, and used either a number 7 or 8 PROM; the later type has 3 edge connectors at the top of the board, and uses a number 9 or 10 PROM.

EARLY TYPE

Remove the PROM; it is no longer needed. Cut the connections between U22-6 and A11; U22-12 and A7; U16-3 and A14; U16-1 and A13. Add jumpers to connect U22-6 to A14; U22-12 to A13; U16-3 to A11; and U16-1 and A7. The original connection of A14 to U16-3 continued on the other side of the board to the PROM; this connection is unnecessary after the modification, but may be left intact with no ill effects. All cuts should be made in close proximity to the ICs with which they are associated.

LATER TYPE

Remove the PROM; it is no longer needed. Cut the connections between U6-6 and A5; and U19-2 and A7. Jumper U19-2 to A5; and U6-6 to A7. Change the jumpers on the top side of the board above U6 to obtain the following connections: U6-13 to pullup; U6-11 to ground; U6-12 to pullup; and U6-10 to ground. The jumpers above U7 need not be changed since U7 is not in use; U7 may be removed if desired.

It is suggested that the board be marked with a tag bearing the F77C address; this will supply a positive indication that the board is not usable in a 9FFC system, thereby possibly saving much confusion at a future date.

DCB-4

The DCB-4 is unlike the BFD-68 controller in that it does not contain firmware, but it does use a number of option jumpers to select different operating features.

To configure the DCB-4 for operation with the 6809 cpu, 4 jumpers on the DCB-4 must be changed.

Cut W1-8 and W2-6
Jumper W2-5 and W2-8

APPENDIX E

```
1:APPNDE.TXT SSB 6809 MACRO ASSEMBLER 2.1 THU OCT 23 1980 @ 16:25:15
MON69   APPENDIX E

                                NAM    MON69
                              * SERVICE INTERRUPT SAMPLE PROGRAM
                              * THIS PROGRAM RINGS THE BELL ON THE TERMINAL
                              * REPEATEDLY UNITL AN X IS TYPED ON THE
                              * KEYBOARD.  THE SERVICE INTERRUPT IS USED TO
                              * ACCESS ROUTINES WHICH PERFORM THE BASIC
                              * FUNCTIONS REQUIRED.
                    F814    A RESTRT EQU    $F814
                    F80A    A OUTCH  EQU    $F80A
                    F808    A INCHK  EQU    $F808
                    F806    A INEEE  EQU    $F806
                    DFCC    A SVCO   EQU    $DFCC
0100                                ORG    $100       THE MAIN PROGRAM
0100 113F                   START   SWI3               RING THE BELL
0102        00      A               FCB    BELL
0103 113F                           SWI3               WAIT SOME TIME
0105        01      A               FCB    WAIT
0106 32     7F      A               LEAS   -1,S        MAKE SOME WORK SPACE
0108 113F                           SWI3               CHECK KEYBOARD
010A 35     02      A               PULS   A
010C 81     58      A               CMPA   #'X        DID WE GET AN 'X ?
010E 26     F0   0100               BNE    START      LOOP 'TIL WE DO
0110 6E     9F F814 A               JMP    [RESTRT]   BACK TO THE BARN
                              * THE NEXT SECTION OF CODE CONTAINS THE
                              * ROUTINES FOR BELL, WAIT, AND KBCHK
0700                                ORG    $700       THE UTILITY ROUTINES
0700 86     07      A RING  LDA    #7         BELL CODE
0702 AD     9F F80A A       JSR    [OUTCH]    SEND IT TO THE TERMINAL
0706 3B                     RTI
0707 8E     FFFF    A DELAY LDX    #-1        LOAD A BIG NUMBER
070A 30     1F      A DELAY1 LEAX  -1,X       COUNT IT DOWN
070C 26     FC   070A       BNE    DELAY1
070E 3B                     RTI
070F AD     9F F808 A CHECK JSR    [INCHK]    CHECK KEYBOARD FOR CHARACTE
0713 27     06   071B       BEQ    NOPE
0715 AD     9F F806 A       JSR    [INEEE]    CHAR READY, GET IT
0719 A7     6C      A       STA    12,S       SAVE CHARACTER ON STACK
071B 3B                 NOPE RTI
```

```
51
52                          * THE FOLLOWING CODE CONTAINS THE SETUPS
53                          * FOR THE MONITOR SWI3 FUNCTION
54
55A 0E00                             ORG     $E00
56
57A 0E00        0700     A TABLE    FDB     RING      POINTER FOR BELL ROUTINE
58A 0E02        0707     A          FDB     DELAY     POINTER FOR WAIT ROUTINE
59A 0E04        070F     A ETABLE   FDB     CHECK     POINTER FOR KBD CHECK
60
61              0000     A BELL     EQU     0         EQUATE EACK ARGUMENT TO
62              0001     A WAIT     EQU     1         ITS POSITION IN THE TABLE
63              0002     A KBCHK    EQU     2
64
65A DFCC                            ORG     SVCO      SERVICE ORIGIN
66
67A DFCC        0E00     A          FDB     TABLE
68A DFCE        0E04     A          FDB     ETABLE
69
70                                  END
```

No Errors detected during this assembly
No Warnings noted during this assembly